Assignment 1

DS405B - Practical Deep Learning from Visual Data

Aseem Behl
School of Business and Economics
University of Tübingen
Sunday, 1. May 2022

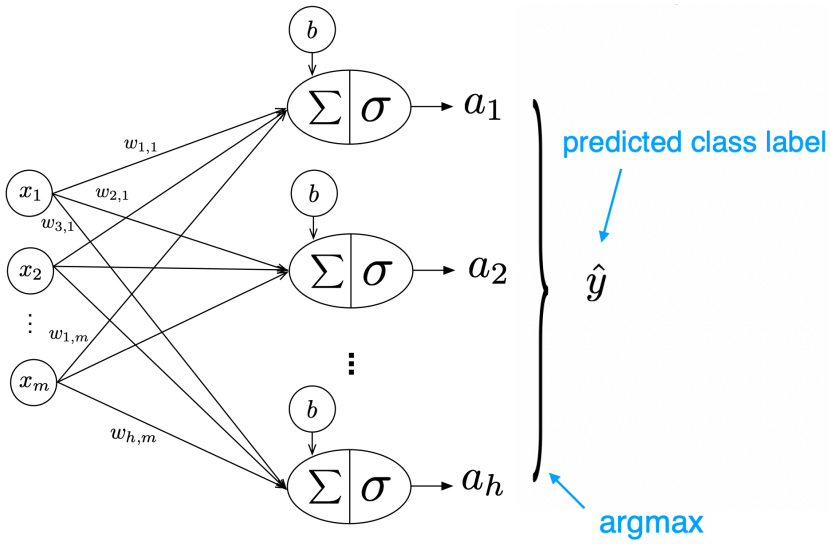
Part 2 - Multi-class Classification

In the second part of this assignment, you will implement, train and evaluate several single layer neural networks to predict the correct category of a previously unobserved plant given certain attributes of the plant as input. We will employ the *Iris Data Set* for this task. The dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The goal is to predict the class of iris plant (Iris Setosa, Iris Versicolour or Iris Virginica) using its 4 attributes: sepal length, sepal widths, petal length, petal width. This is an exceedingly simple domain, however, it is still a useful exercise to develop better understanding of linear, logistic and softmax regression methods from the perspective of neural networks. In the companion notebook, the dataset is prepared for you by importing it from the dataset file, mapping the textual categories (Iris Setosa, Iris Versicolour or Iris Virginica) to numerical class labels (0, 1 or 2) and splitting of the dataset into training (80%) and test (20%) datasets.

Furthermore, the input features are normalised to have zero mean and unit standard deviation. Normalisation (also called standardisation or rescaling) of input features is an important step before training machine learning models. Intuitively, we can think of a scenario of gradient descent with features being on different scales. In such a scenario, certain weights may update faster than others since the feature values play a role in the weight updates. Thus, it is helpful to bring them to same scale before inputting to the model.

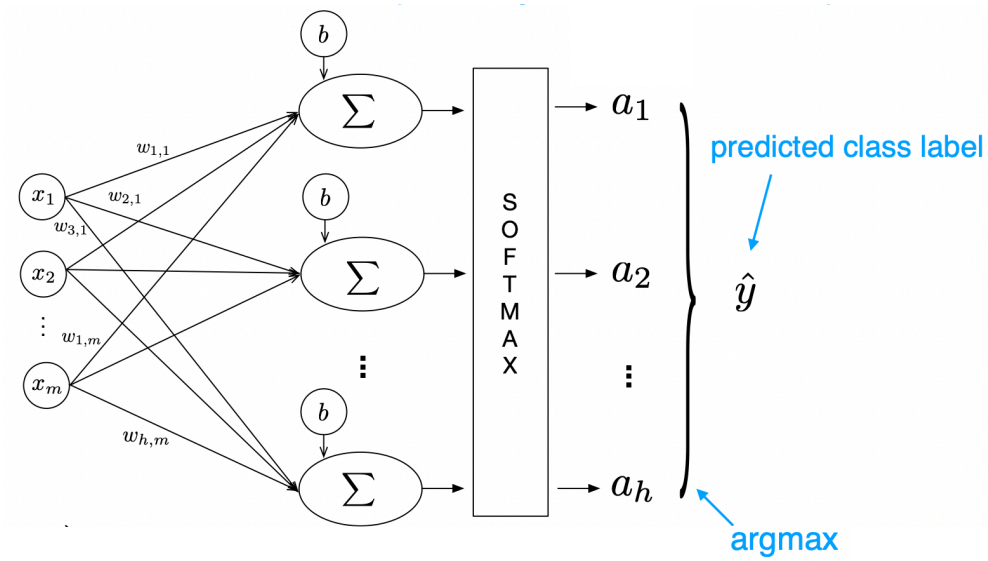
The multi-class classification can be tackled with several different network architectures and loss functions. Your task in this exercises is to implement and train the following **6** different models. After training on the train set, you will also evaluate each model by computing the classification accuracy on the test set.

1. **Logistic Regression for Multiclass Classification:** One vs. all (also called one vs. rest) approach provides a way to leverage binary classification for multi-class classification. Given a classification problem with h possible solutions, a one-vs.-all solution consists of h separate binary classifiers—one binary classifier for each possible outcome. During training, the model runs through a sequence of binary classifiers, training each to answer a separate classification question. For example, given an example of Iris Setosa, 3 different classifiers might be trained, two seeing the example as a negative example (not an Iris Setosa) and one seeing the example as a positive example (an Iris Setosa). This approach is fairly reasonable when the total number of classes is small, but becomes increasingly inefficient as the number of classes rises. We can create a significantly more efficient one-vs.-all model with a neural network in which each output node represents a different class. The following architecture suggests this approach:



As we discussed in the lecture (week 4), the above illustrated architecture is equivalent to h logistic regression models one for each of the h binary classification problems. σ here refers to the logistic sigmoid function. The network is trained using a binary cross entropy loss for each output node. And finally, the predicted class $\hat{y}$ is the output neuron a_i with the largest activation value.

2. **Softmax Regression with custom implementation of Cross Entropy Loss:** Softmax extends the idea of logistic regression into a multi-class problem. That is, Softmax assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would. Therefore, for the second model you will implement the architecture as shown below.



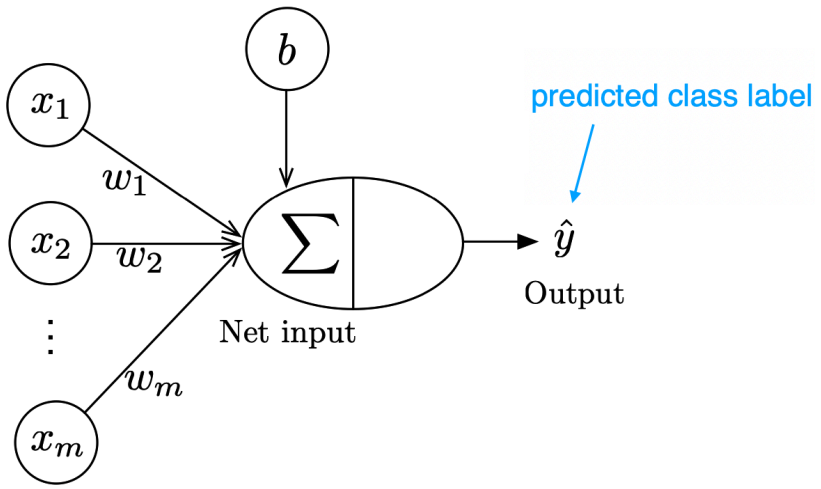
For this model, please implement your own versions of one-hot encoding function, softmax and cross-entropy loss function.

3. **Softmax Regression with torch.nn.functional.nll_loss:** Implement the same model architecture as 2, however this time make use of the `torch.nn.functional.nll_loss` function. Note that `nll_loss` takes `log(softmax)` as input.

4. **Softmax Regression with torch.nn.functional.cross_entropy:** Implement the same model architecture as 2, however this time make use of the `torch.nn.functional.cross_entropy` function. Note that `cross_entropy` loss takes *logits* as input.

5. **Softmax Regression with Mean Squared Error Loss:** For multi-class classification, we can also use the use mean squared error during training. The technique is to use one-hot encoding for the ground truth class labels, and `softmax()` activation on the output nodes. For example, if there are 3 classes then a target label might be (0, 1, 0) and a predicted output of the model might be (0.10, 0.70, 0.20), the squared error would be (0 – 0.10)^2 + (1 – 0.70)^2 + (0 – 0.20)^2.
   Therefore, for this model, you will implement the same model architecture as 2, however this time use the Mean Squared Error Loss function.

6. **Linear Regression with Mean Squared Error Loss:** So far all the previous models you implemented in this exercise predict the probabilities of the example x being from class 0, 1 or 2. An alternative to this is to directly regress the label $\hat{y}$ as a continuous value between 0 and 2 (if we have 3 categories). The architecture below suggests this approach:



As we are not regressing the probabilities, we can get rid of the logistic sigmoid function. Furthermore, as we are regressing a real valued output, it would make sense to use Mean Squared Error Loss for this model. Finally, the discrete class label can be computed by rounding the regressed prediction to the closest integer value.

This part of the assignment is worth **10 points** and is due on **Monday, May 30, at 08:00 AM CET**.

In order to receive credit for the assignment, please be prepared to present your solutions to selected exercises from the assignment or other similar exercises in the lecture on **Wednesday, June 01, 2022**.

Submission Steps

1. Download the starter notebook assignment_1B_2022.ipynb and the dataset file iris.data from ILIAS.

2. Please upload the notebook and the dataset file to DS405B folder on your Google Drive.

3. Then, open the notebook file with Google Colab by right-clicking the *.ipynb file.

4. You can save your work in Google Drive (In Colab click “File” -> “Save”) and resume later.

5. **Run all cells, and do not clear out the outputs, before submitting.**

6. Download the saved notebooks to your computer and upload it as submission to ILIAS.